DESCRIPTION

IMPROVED INVERSION CALCULATIONS

The present invention relates to a method of performing an inversion operation and to apparatus for performing an inversion operation.

Elliptic Curve Cryptography (ECC) involves the use of calculations on an elliptic curve relationship over GF(p) or GF(2^n)and requires the multiplication of long integers which are carried out repeatedly during the implementation of, for example, public key algorithms in cryptographic processors.

Typically, the multiplication operations must be carried out many hundreds of times to complete an encryption or decryption operation, and so it is important that the cryptographic devices that perform these operations execute the long multiplications quickly using a high speed multiplier.

ECC calculations require also an inversion calculation, i.e. the calculation of $\mathbb{Z}^{-1}$, such that the product $\mathbb{Z}.\mathbb{Z}^{-1}$=1 mod $\mathbb{N}$. Every point addition and point doubling calculation requires such a calculation. The present algorithms are computational intensive.

Another way is working in the so-called Projective Space. This postpones the inversion calculation to the end and has to be done only once, but the trade-off is that the number of multiplications is largely increased.

Increasingly, such cryptographic algorithms are used in electronic devices for example smart cards, and in these applications processing capability and power consumption is severely limited.

One conventional calculation method is the binary GCD system which works with pairs of auxiliary variables. One pair is reduced in size by dividing by 2 when even, or by subtracting when odd.

However, in the GCD system often it is necessary to correct the operation on the other pair by the addition of half of the modulus.

Another conventional calculation method is the Kaliski system which again uses two pairs of auxiliary variables, of which one pair is reduced by dividing by 2 when even, or by subtracting when odd.

However, in this system, any required correction is delayed to the second stage.

It is therefore an object of the present invention to provide a more efficient inversion operation.

It is also an object of the present invention to provide a inversion process with fewer operations.

It is also an object of the present invention to provide an inversion operation which is completed faster than in conventional systems.

According to one aspect, the present invention provides a method of performing an inversion operation in a cryptographic calculation with at least two auxiliary variables, the method comprising shifting a variable, then effecting a reduction by subtracting that variable from a larger variable.

One advantage of the present invention is that most operations are only done on the Most Significant Words of the auxiliary variables. After a number of such computations, a number of multiplications are done on the complete auxiliary variables, which are simpler.

These advantages result in the number of necessary operations being reduced as compared to conventional methods, thereby ensuring that the calculations can be effected more quickly.

Thus a significant benefit provided by the present invention is that the time taken to complete the entire calculating operation is reduced.

Moreover, the degree of security afforded by the method of the present invention is maintained as compared to conventional cryptographic methods.

Preferably, the method comprises four auxiliary variables being U, V, R and S having the invariances:-

$|S.V-R.U| = N$

$S.Y = U \bmod N$

$R.Y = V \bmod N.$

Preferably, the method operates with the Most Significant Words of the variables.

Thus an advantage of the present invention is that the calculation operations are effected faster.

According to another aspect, the present invention provides a computer program product directly loadable into the internal memory of a digital computer, comprising software code portions for performing the method of the present invention when said product is run on a computer.

According to another aspect, the present invention provides a computer program directly loadable into the internal memory of a digital computer, comprising software code portions for performing the method of the present invention when said program is run on a computer.

According to another aspect, the present invention provides a carrier, which may comprise electronic signals, for a computer program embodying the present invention.

According to another aspect, the present invention provides electronic distribution of a computer program product, or a computer program, or a carrier of the present invention.

According to another aspect, the present invention provides apparatus for performing an inversion operation in a cryptographic calculation with at least two auxiliary variables, the apparatus comprising means to shift a variable, and means to effect a reduction by subtraction or addition of that variable from a larger variable.

The method and apparatus of the present invention is applicable to calculations over GF(p), GF(2^n) and also long-integer division.

In order that the present invention may more readily be understood, a description is now given, by way of example only, reference being made to the accompanying drawings, in which:-

Figure 1 is a block diagram of an application of the invention in a smart card;

Figure 2 is a schematic drawing of an inversion operation embodying the present invention;

Figure 3 is a hardware implementation of the present invention;

Figure 4 is a further detailed hardware implementation of the present invention;

Figure 5 is a schematic drawing of another inverse operation of the present invention;

Figure 6 is a schematic drawing of another inverse operation of the present invention;

Figure 7 is a schematic drawing of a further operation of the present invention.

Figure 1 shows a block diagram of a hardware implementation of the present invention incorporating a smart card 50 with the following components:

- Microcontroller 51 for general control to communicate with the outside world via the interface. It sets pointers for data in RAM/ROM and starts the coprocessor.
- Interface to the outside world, for contact with smart cards e.g. according to ISO-7816-3.
  - A Read Only Memory (ROM) 52 for the program of the microcontroller.
  - A Programmable Read Only Memory (Flash or EEPROM) 53 for the non-volatile storage of data or programs.
  - RAM 54 for storage of volatile data, e.g for storage of intermediate results during calculations.
  - Coprocessor 55 dedicated to perform special high-speed tasks for ECC or RSA calculations. When a task is ready, control is returned to the microcontroller.

In a variant, the present invention is implemented in software with a microprocessor, ALU to provide add, subtract, shift operations with programming of the controller to provide control logic, and degree detection by shift registers. 2

There is shown in Figure e an inversion operation of the present invention which is described below.

Thus this method of calculation over GF(p) involves the operation

$R = Y^{-1} \bmod N$

having four auxiliary variables U, V, S and R, with

$U = Y$

$V = N$

$S = 1$

$R = 0,$

U and V always being positive.

The degree of an auxiliary variable is the number of relevant bits to represent it. Thus for example, if U = 111100

then the degree of U = dU is 6;

and, if V = 001110,

then the degree of V = dV is 4.

The operation involves taking:

$B = dU - dV$ (Step S1);

and, if b<0, then performing the operations (Step S2, S3):-

(swap U, V)

(swap R, S)

(swap dU, dV)

$b = -b$

then $U = U-2^{b}.V$

$S = S - S^{b}.R$

and if (U< 0)

then (Step S4) $U = - U$

$S = -S,$

if (R<0), then $R = R+N$

if (R>N), then $R = R-N$.

Thus the following invariants hold after each loop iteration:

$\gcd(U,V) = \gcd(Y,N)$

$SY = U \bmod N$

$RY = V \bmod N$

$|SV-RU| = N.$

In every step, either the degree of U is decreased or the degree of V. Therefore U and V become smaller and smaller, until in the last step U becomes 0 ($U=2^bV$).

Since U= 0, the invariance gcd(U,V)=gcd(Y,N) implies V=gcd(Y,N)=1, since Y and N are relative prime.

Then RY=1 mod N or $R = Y^{-1}$ mod N.

When U= 0, -N<R<2N,

giving at most one correction step namely: either adding or subtracting N.

In practice, R appears always to be smaller than N, so that subtraction of N never occurs.

Also, |SV|<2N and |RU|<2N temporary.

Since they are all integers,

|S|<2N;

|V|<2N;

|R|<2N;

|U|<2N.

For these variables, only one bit more than N requires representing them. For S and R, a sign-bit is needed too.

Figure 2 shows the hardware implementation of the method of the present invention.

Registers 10, 11, 12 and 13 hold variables U, V, S, R. The adders 14, 15 perform addition, subtraction, negation and mod 2 additions. V and R can be shifted over b bits. The control logic 16 controls the process. There are two degree detectors 17,18, one for U and one for V. The dSubtractor 19 gives the difference (b).

Initially, Y is loaded into U, N into V, S is set to 1 and R to 0.

Then the process is started.

When b<0, U and V exchange their contents, S and R do the same, and b is negated.

Both adders are set to subtraction and the shifters are set to shift over b bits. Then the subtraction is performed. When U is negative, the adders are set to negate both U and S.

The process is done as long as $U \neq 0$.

When U=0 and R<0 or R>N, S is loaded with N. Then either R+N or R-N is calculated.

Normally, the operands consist of a number of words. However, in a variant, the calculations can be speeded up by using only the Most Significant Word two of the variables and 4 auxiliary variables with the size of 1 word, while keeping the invariances valid. It saves also chip area and power. The result is used as an estimator for the subsequent calculation on the whole operands.

Figure 3 shows the more detailed hardware implementation. Registers 30 to 35, each with a 1 word capacity, hold U_H,V_H,uu, uv, vu and vv.

U_H and V_H are initially loaded with the Most Significant Word of U and V.

$U = uu.U_0 - uv.V_0$

$V = vu.U_0 - vv.V_0$

$S = uu.S_0 - uv.R_0$

$R = vu.S_0 - vv.R_0$

uu,uv vu and vv are words of convenient size.

The operation starts with uu=1, vv=-1 and uv=vu=0,

$U_0 = Y$;

$V_0=N$;

$S_0=1$;

$R_0=0$.

Assume that the equations are still correct after a number of steps. After the next calculation, the equations are still correct. Since they are correct in the beginning, they remain correct.

When calculating $U' = U-2^bV$ and $S'= S-2^bR$, then choose:

$uu'=uu-2^bvu$

$uv'=uv-2^bvv$

$vu'=vu$

$vv'=vv$.

When it is necessary to calculate $U'=U+2^bV$ and $S'=S+2^bR$, then choose:

$uu' = uu + 2^b vu$

$uv' = uv + 2^b vv$

$vu' = vu$

$vv' - vv$.

When required, swap uu and vu, uv and vv.

This swaps U and V as well R and S.

To update the operands, start with loading U_H with MSW of U and V_H with the MSW of V. Then,

uu=1,vv=-1 and uv=uv=0.

Then a number of calculations are done, the amount depending on the size of the words and how many useful bits are left over.

Since V_H is shifted, it is supplemented with zeros instead of the (unknown) right bits so U_H and V_H become smaller and smaller. The operation is halted when there are almost no bits left. Also the determination of the sign become incorrect.

Then calculate U, V, S and R by means of uu...vv and $U_0 \ldots S_0$.

This gives new reduced values of U and V, which still obey the invariance.

Then set U_0 to U, V_0 to V and the same for S_0 and R_0. Again set uu=1, vv=-1 and uv=vu=0.

Then repeat the procedure. Every time U and V become smaller and smaller, until they fit in the U_H and V_H registers.

Then the calculation is no longer an estimation, but an exact calculation and it ends with the correct result. Finally, only R has to be recalculated to find Y^{-1}

In a variant to the method of Figures 1 to 4, the calculation method allows negative values for U and V and removes the correction step when U is negative (see Figure 5).

The degree of positive numbers is the number of bits after removing all leading zeroes and the degree of negative numbers is the number of bits after removing all leading ones.

Again, the auxiliary variables are:

```
        U=Y;
        V=N;
        S=1;
        R=0;
    while (U≠0) and
    if (b<0) then effect:
        {swap (U,V); swap (R,S) swap(dU,dV); b=-b};
    if (Sign(U)=Sign(V))
    then effect
        {U=U-2^b.V;S=S-2^b.R;}
        Else
        {U=U+2^b.V; S=S+2^b.R;}
        dU=degree(U);
        if (R<0), then R=R+N;
    if (R>N) then, R=R-N.
```

Figure 6 shows a second embodiment which is a calculation method over $GF(2^n)$, the major differences being:

α is the variable of the polynomials, U, V, S and R;

N is the irreducible polynomial;

the algorithm is simpler since there are no negative values and there is only a mod 2 addition.

Thus with

```
        U=Y;
        V=N;
        S=1;
        R=0;
    while (U>0)
        b=dU-dV
        if (b<0) {swap(U,V);swap(R,S); swap(dU,dV); b=-b;}
        U=U⊕α^b.V;
        S=S⊕α^b.R;
```

d=degree(U);

if (R>N) R=R⊕N.

Thus, initially, Y is loaded into U, N into V, S is set to 1 and R to 0.

Then the process is started (Steps S10-S12).

When b<0, U and V exchange their contents, S and R do the same and b is negated.

Both adders are always set to add mod 2. The shifters are set to shift over b bits. Then the addition is performed.

The process is done as long U≠0

When U=0 and R=R>N, S is loaded with N, then R ⊕ N is calculated.

Figure 7 shows a third embodiment which is a calculation method for long-integer division, the major differences being:

Initially, X is loaded into U, Y into V, S is set to 0 and R to 1.

When U>0, then the UV–adder is set to subtraction and the RS-adder to addition, or the reverse is done, as appropriate. The shifters are set to shift over b bits. Then the addition/subtraction operation is performed.

The process is done for as long U≠0 and b≥0.

When the process is ready and U<0, then b is set to 0. Then one addition/subtraction is performed (U=U+V; S=S-R).

Then U is the remainder R' and S is the quotient Q, X = Q.Y+R' with 0≤R'<Y.